5. Emiliano

Program Name: Emiliano.java **Input File: emiliano.dat**

Emiliano loves morphological derivation and the most fundamental and fascinating to him is compound words. He is building a program to identify every compound word in a given dictionary! Given a list of words in this dictionary and a list of test words, determine which of the test words are compound words. A test word is considered a compound word if it is completely made up by exactly two dictionary words. But if a single test word can qualify as a compound word in multiple ways, it is known as a multi-compound word. For example, the word NOTABLE is a multi-compound word:

```
NOTABLE is a compound word of NO and TABLE.
NOTABLE is a compound word of NOT and ABLE.
```

Thus, the test word notable will count as two compound words.

For each of the test words, determine if it is a compound or multi-compound word made up by a pair of dictionary words. For each way that the test word qualifies as a compound, increment the count of the number of compound words. In the case of multi-compound words, each compound pairing is counted.

Input: The first integer will represent the number of data sets to follow. The next line will be an integer, a, representing the number of dictionary words. The next a lines are single words (with no spaces) representing the dictionary words, in an arbitrary order. The following line will be an integer, b, representing the number of test words. The next b lines are single words (with no spaces) representing the test words.

Output: For each test word, determine if it is a compound word, and if it is, add one to the count. For each data set, print the number of compound words (and multi-compound word pairings).

Assumptions: Language is very complex, and thus the number of words in the dictionary can be very, very large. There is a 3 minute execution time limit and a 1GB space limit. Note the empty string is not a dictionary word. Prefix dictionary word and suffix dictionary word need not be unique.

Sample Input:

```
3
3
car
Bar
STAR
5
barcar
CARSTAR
BarStar
Starsnbars
Nobarcar
3
a
aa
beans
4
aaa
abeans
abeansa
beansa
2
no
yes
1
yesno
```

Sample Output:

```
3
4
1
```